ЭФФЕКТИВНОЕ СОЗДАНИЕ ТЕКСТОВЫХ ИНДЕКСОВ

Веретенников А.Б.
e-mail: abvmt@e1.ru

Задача поиска в больших массивах текстов весьма актуальна в связи с постоянным ростом объемов информации. Для этого применяются в основном инвертированные файлы [1] и их аналоги. Существует много поисковых систем в Интернет, созданных на основе инвертированных файлов. Однако в инвертированные файлы сложно добавлять новые данные.

В [2] было представлено String-B дерево, эта структура данных представляет собой B-дерево [3], в котором ключи могут иметь произвольную длину.

В [4] была введена в рассмотрение новая структура данных CLB-дерево, которая представляет собой B-дерево, обладающее способностью хранить в себе как ключи, так и их значения имеющие произвольную длину. Эта структура данных может быть использована для индексации текстов. В качестве ключей могут выступать слова, или их формы, а в качестве значений ключей – список вхождений этих слов в индексируемых текстах.

Эта структура данных была использована автором в качестве одного из компонентов новой структуры данных – CLB-индекс.

CLB-индекс можно рассматривать как эффективную комбинацию нескольких CLB-деревьев, и специальным образом организованных небольших таблиц в оперативной памяти.

Также CLB-индекс включает в себя такие необходимые компоненты как список документов, хранящий в себе описания обработанных документов и при необходимости репозитарий, хранящий в себе содержимое этих документов в сжатом виде.

При создании индекса активно используется анализ морфологии языка [5]. За счет использования морфологического анализа можно значительно увеличить скорость создания индекса, за счет оптимизации кэширования. Кроме того, учет морфологии языка необходим для повышения точности поиска.

Основное достоинство CLB-индекса в том, что в него можно легко добавлять новые данные, в отличие от инвертированных файлов,

которые обновлять сложно. По сути, вычислительная сложность обновления CLB-индекса – линейная, в то время как у инвертированных файлов – экспоненциальная. Кроме того, по скорости поиска CLB-индекс не уступает инвертированным файлам.

Следует отметить, что производительность CLB-индекса (уже весьма высокая на современных носителях) значительно возрастет при применении SSD накопителей, которые появляются в настоящее время, в то время как на производительность инвертированных файлов, это окажет значительно меньшее влияние.

При организации CLB-дерева мы вводим два типа страниц – страницы B-дерева, в которых хранятся ключи и кластеры, в которых хранятся значения соответствующих ключей.

Пусть d – доля известных слов в текстах, т. е. тех, которые входят в словарь морфологического анализатора, R – текущее количество слов в CLB-дереве, H – размер страницы B-дерева, K – размер кластера.

Справедливы следующие теоремы:

Теорема 1. *Вставка N слов в CLB-индекс потребует* $O(d \cdot N/K + (1-d) \cdot N \cdot (1 + log_H((1-d) \cdot (R+N))))$ *обращений к внешней памяти.*

Теорема 2. *Поиск фразы из N слов в CLB-индексе потребует* $O(N \cdot (log_H((1-d) \cdot R) + occ/K)$ *обращений к внешней памяти, где occ – количество вхождений данных слов в текстах.*

При поиске с помощью инвертированных файлов мы получаем вычислительную сложность $O(N \cdot (occ/K))$.

Заметим, что значение $log_H((1-d) \cdot R)$ очень мало по сравнению с occ/K, за счет параметра H, таким образом, по скорости поиска CLB-индекс не уступает инвертированным файлам.

Для поиска с помощью CLB индекса автором разработана система, которая имеет модульную структуру. За счет различных модулей обрабатываются документы различных форматов (обычный текст, RTF, PDF, HTML, CHM, ...) и архивы (ZIP, RAR, CAB, ...). При необходимости система автоматически определяет кодировку текста, с использованием анализа морфологии.

25 гигабайт текста (300 000 файлов) индексируется за 3 часа, из них 2 часа – чтение текстов с диска и 1 час – создание индекса.

Указанный эксперимент проводился на машине с процессором Core 2 Duo E6700 и 4-мя гигабайтами оперативной памяти.

В настоящее время автор концентрирует свое внимание на проблеме поиска похожих документов, с целью добавить подобную функциональность в свою систему.

Список литературы

[1]. *Prywes, N. S., Gray, H. J.* The organization of a Multilist-type associative memory. IEEE Trans. on Communication and Electronics, 68 (1963), 488-492.

[2]. *Ferragina, P., Grossi, R.* The string B-tree: a new data structure for string search in external memory and its applications. Journal of the ACM, 46, 2 (1999), 236-280.

[3]. *Bayer, R., McCreight, E.* Organization and maintenance of large ordered indexes. Acta Informatica 1, 3 (1972), 173-189.

[4]. *Веретенников А. Б., Лукач Ю. С.* Еще один способ индексации больших массивов текстов. Известия Уральского государственного университета. Серия “Компьютерные науки”, 2006, №43. с. 103-122.

[5]. *Лукач Ю. С.* Быстрый морфологический анализ флективных языков. Международная алгебраическая конференция: К 100-летию со дня рождения П. Г. Конторовича и 70-летию Л. Н. Шеврина. Тез. докл. Екатеринбург: Изд-во Урал. ун-та, 2005, с. 182-183.